AllThingsTalk RDK Kit Onboarding

Start up Guide with Proximus LoRa4Makers and CloudEngine

Sensitivity: Unrestricted

Contents

Sensitivity: Unrestricted

1 Resources & Setting up The Prerequisites

- AllThingsTalk (ATT) LoRaWAN Rapid Development Kit
- Device UID (on ATT Kit box. If no devEUI on your box, you can ask one at enco@proximus.com)
- ATT RDK connection schema: https://support.sodaq.com/Boards/Mbili/
- Arduino IDE : https://www.arduino.cc/en/Main/Software
- Setup the SODAQ MBili Board support in the Arduino IDE:
    - Go to https://support.sodaq.com/getting_started/
    - Use the URL for the "stable AVR" channel
    - Once you added the URL to your IDE settings according to the instructions on this page, follow the installation instructions to install the board support via the Board Manager
    - Once the board support files are installed, make sure you select the "SODAQ Mbili 1284p 8Mhz Optiboot at 57600 bauds". If your board is already connected via USB to your computer, make sure to select the right port.
- Install ATT SDK in Arduino IDE:
    - Download the AllThingsTalk Arduino SDK
    - To install, unzip the downloaded file and open your Arduino IDE
    - Under Sketch > Include library > Add .ZIP library and select the arduino-lorawan-sdk-master file
- Install ATT RDK in Arduino IDE:
    - Download from https://github.com/allthingstalk/arduino-lorawan-rdk
    - Install in Arduino IDE same as for SDK above

2 Create & activate your Proximus API Solutions account

Go to http://proximusapi.enco.io

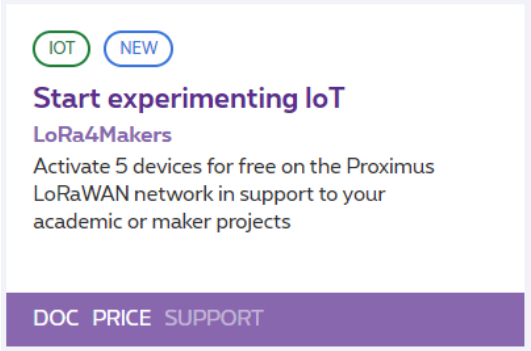
To register a new account, click on Login, then Register and follow the procedure.

2.1 Subscribing to LoRa4Makers

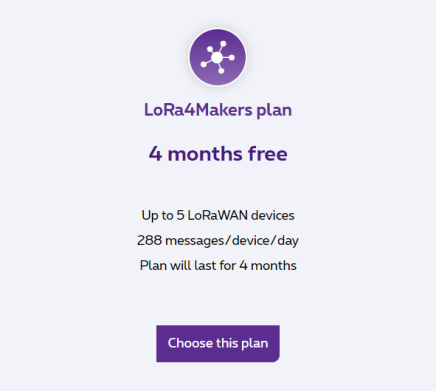
Once registered, log in the platform. To onboard new LoRa devices on the Proximus LoRaWAN network, you will need to get a subscription. Via the API Solutions platform, you can get free access to our network for 4 months for up to 5 devices. This subscription is called "**LoRa4Makers**".

From the API Solutions website, click on "Price" under the LoRa4Makers tile:

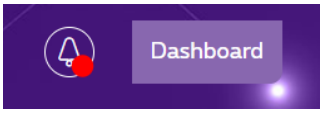
Sensitivity: Unrestricted



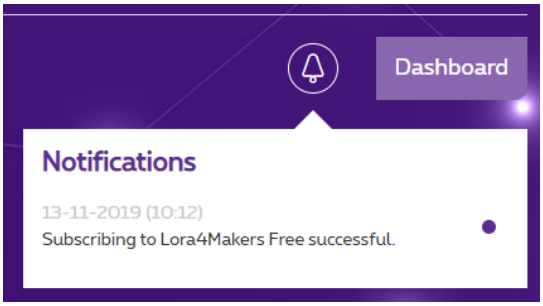
Then click on “Choose this plan”



You can follow the status of your registration via the notifications on the top of the window:



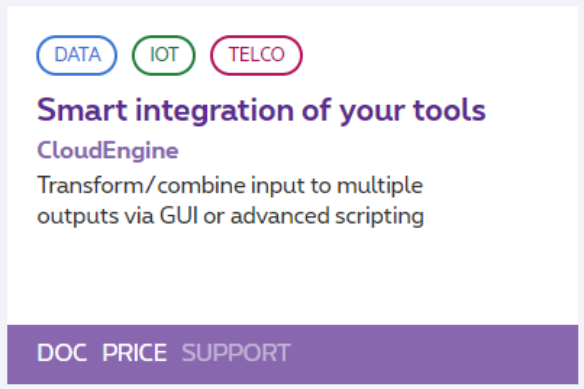
Make sure you receive the following notification:



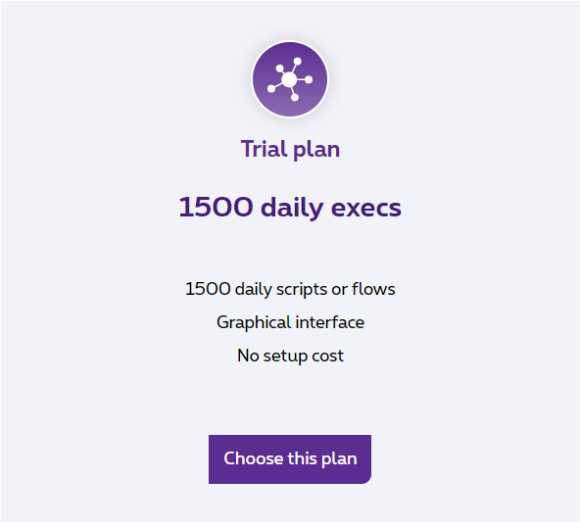
2.2 Subscribing to CloudEngine

Any data sent by your LoRa RDK will be sent to the **Proximus CloudEngine**. The CloudEngine is a serverless scripting environment that lets you define simple and complex flows, triggered by incoming data. Simple flows can be designed visually while more advanced data processing will use a scripting mode based on EnCoScript, a language very close to Javascript.

From the API Solutions website, click on “Price” under the CloudEngine tile:



On the pricing page, select the Trial plan:



Watch for the subscription status progress in the notification area.

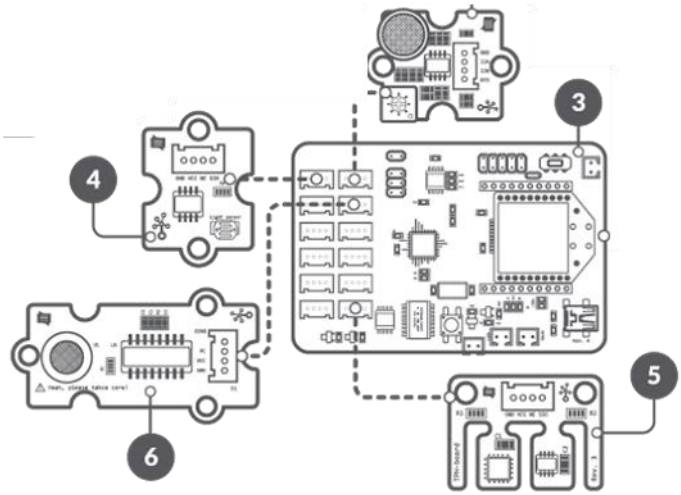
3 Implementing your first AllThingsTalk project

3.1 Preparing the hardware

For your first project, we will implement the “environmental-sensing” scenario which makes use of the following sensors, available in your ATT RDK:

- TPH (temperature, pressure, humidity) sensor
- Air quality sensor
- Light sensor
- Sound level sensor

Sensitivity: Unrestricted



- Attach LoRa™ module & antenna
- Connect air quality sensor to A0/A1
- Connect the light sensor to A2/A3
- Connect noise sensor to A4/A5
- Connect TPH sensor to SCL/SDA (I2C socket)
- Connect Sodaq Mbili to your computer through USB
- Turn power switch ON

3.2 Creating the device in LoRa4Makers

Back on proximusapi.enoc.io, click on the “Microsite” link under the LoRa4Makers tile.

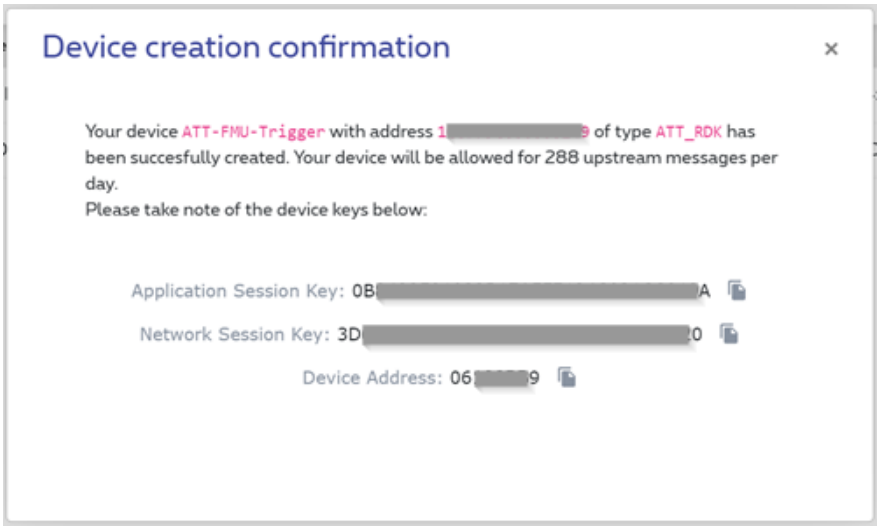
Mind that an inactivity timeout might kick in and results in an error message when loading the microsite. Should this happen, log out from the site (via the “Dashboard” menu), then log back in and retry.

1. Click on + button to create a new device
2. Provide a unique device ID (name) and device Unique Identifier (devEUI). Use the devEUI printed on the side of your ATT kit. If not such label is available, you can ask us (enco@proximus.com) for a unique devEUI.
3. Associate at least one tag to the device. Press enter or tab after typing a tag name.
4. Choose the device type “AllThingsTalk RDK”
5. Click Add

A confirmation windows will appear with all data you may need to configure your hardware device, namely:

- Device address
- Application Session Key
- Network Session Key

Sensitivity: Unrestricted



3.3 Configure your ATT board

Connect the kit to your computer over USB and make sure you have followed all instructions from sections 1 and 3.1.

In the Arduino IDE, select “Examples” from the File menu, and down the list select “Arduino-lorawan-rdk-master”, and from there : “environmental-sensing”.

At the beginning of the project file, you need to select “CONTAINERS” as your preferred method for sending date. The lines should look as below:

```
// Select your preferred method for sending data
#define CONTAINERS
//#define CBOR
//#define BINARY
```

Next, you need to update the keys.h file, opened in the IDE, with the device address (DEV_ADDR), application session key (APPSKEY) and network session key (NWKSKEY). You have received all of these values when you created your device in LoRa4Makers. If you didn’t note them down, just go back to LoRa4Makers and click on your device.

Note that these data have to be specified in hexadecimal format in the keys.h file. You’ll need to separate each group of 2 characters by a “0x”, as shown below:

```
uint8_t DEV_ADDR[4] = {0x08, 0x1B, 0x3A, 0x74};
uint8_t APPSKEY[16] = {0x07, 0xAC, 0x33, 0x96, 0x24, 0x72, 0xD9, 0xEB, 0xD0, 0x7A, 0xB4,
0x77, 0x75, 0xBA, 0xA8, 0xE5};
uint8_t NWKSKEY[16] = {0xC2, 0x3F, 0xB2, 0xE5, 0xDF, 0xDF, 0x2D, 0x7E, 0x2C, 0x02, 0xBE,
0xF3, 0x73, 0x1E, 0x5A, 0xE0};
```

Save your Arduino project.

Compile the arduino sketch and check for any error (there shouldn’t be any, unless you did not import both ATT SDK and RDK files correctly). Transfer the compiled sketch to your board and open the Serial Monitor (CTRL-Shift-M). Check that the speed setting on the serial monitor is correctly set to 57600. As the compiled sketch is transferred, you should see your board come alive and initialize.

Sensitivity: Unrestricted

The board will send sensor data every 5 minutes. You can change that behaviour by changing the "SEND_EVERY" definition in the schema (don't forget to recompile and transfer to your board). Just don't forget you are allowed 288 message per device per day. As currently configured, the board will send 6 messages every 5 minutes (air quality, temperature, humidity, pressure, light, noise level).

3.4 Create your flow in CloudEngine

Now we need to check the incoming messages on the API Solutions platform.

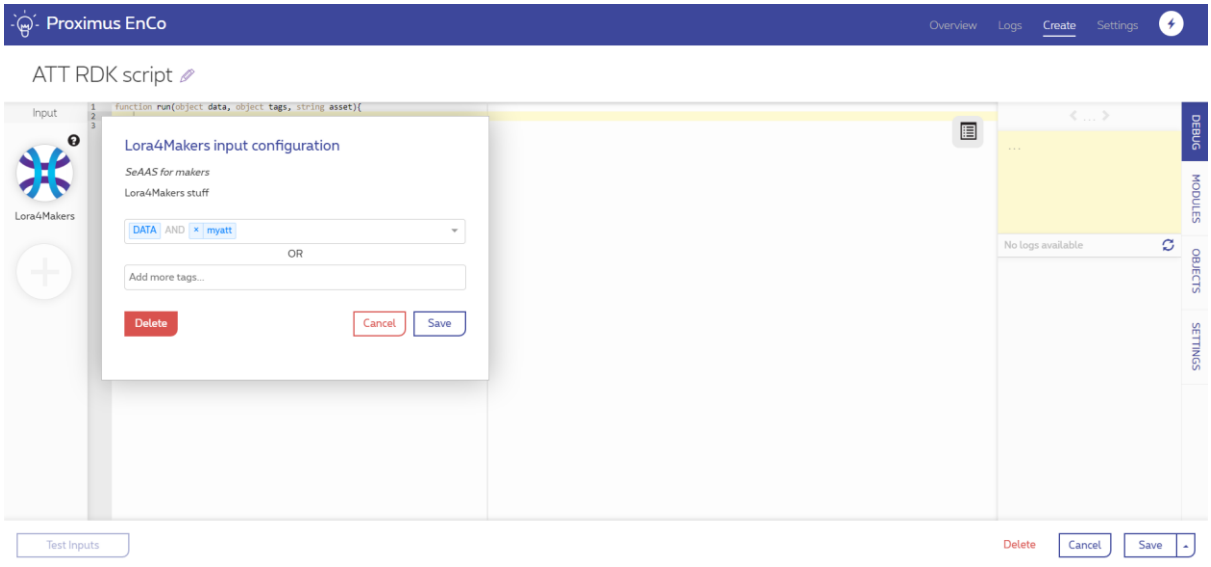
Go to the API Solutions platform, and select the "microsite" link under the CloudEngine tile. Mind that an inactivity timeout might kick in and results in an error message when loading the microsite. Should this happen, log out from the site (via the "Dashboard" menu), then log back in and retry.

CloudEngine allows to create basic flows with a visual editor, or define this flows via scripting. To exploit the data coming in as decoded LoRa messages for AllThingsTalk devices, it is more convenient to use the scripting mode. Documentation on script mode can be found here.

Select "Create", then directly click on "convert to script". Give a name to your new script.

Click on the "+" sign in the input column. This will allow us to define on which inbound event the flow will activate. Select "LoRa4Makers" and close the inbound endpoint selector.

We want this flow to activate only for data coming from this specific LoRa device, so we will add a filter on this input endpoint. To do so, click on the newly added LoRa4Makers inbound endpoint, and next to the "DATA" tag, add a new tag with your LORA device UID, or one of the tag you have associated to your device when creating it. Let's suppose we had associated "myatt" to our device, than my selection would look like this:



Click Save to close the box.

Your LoRa messages will arrive as a array object called "data", with data fields (index) as define below:

The following data fields will always be available in all LoRa messages:

Common data fields

Data field	Description
DevEUI	device unique identifier
Time	The timestamp indicating the moment of the reading
DeviceName	The alias name of the device
DeviceAddress	This is a unique identifier for the device provided by Proximus
FPort	The application port (* or 1-255)
FCntUp	LoRa received packet counter value for uplink
FCntDn	LoRa received packet counter value for downlink
LrrRSSI	LoRa base station Received Signal Strength
LrrSNR	LoRa base station Signal to Noise Ratio
SpFact	LoRa spreading factor
SubBand	LoRa subband used
Channel	LoRa channel used
DvLrrCnt	Amount of lrr's
LrrId	Identifier of the LoRa base station
Tags	JSON array as a string, containing the device-specific tags

And when receiving messages from AllThingsTalk LoRaWAN RDK devices, the following additional data fields may be added, depending on the sensors and schema you have implemented on your board (and making sure you have correctly commented our #define CBOR and uncommented #define CONTAINERS, as explained under section 3.3).

Data field
boolean_value
binary*tilt*value
push*button*value
door*sensor*value
temperature*sensor*value
light*sensor*value
motion*sensor*value
x*motion*, ymotion, z_motion
latitude, longitude, altitude, time
pressure_value
humidity_value
loudness_value
air*quality*value
battery*level*value
integer_value
float_value
binary_value

In our first script example, we will want to:

- convert this “data” array object into a json structure, and
- send that json structure to an external HTTP endpoint via a POST operation.

As we do not have an application HTTP endpoint available, we will use the online tool http://kara.rest which allows to simulate a HTTP endpoint. Go to https://kara.rest/, click on “create” and keep this tab open, we’ll need it in a minute.

Our script will look as below. You can copy/paste the script as is in the script editor, but don’t forget to copy the kara HTTP request URL and replace it in the script below.

```
object debug = create("Debug");
object array = create("Array");
object json = create("Json");


function run(object data, object tags, string asset){
    # Create json with all data from incoming LoRa4Makers data array
    object incomingJson = json.createNewObjectNode();
    foreach (key in data) {
        if (data[key]!=null) {
          incomingJson.set(""+key, ""+data[key]);
        }
    }
    # Create a HTTP object, and send the newly created json to it
    object http = create("HTTP", "https://kara.rest/bin/REPLACE_WITH_YOUR_KARA_BIN", "POST");
    http.setContentType("application/json");
    http.setData(incomingJson);
    http.send();
}
```

Let your sensor send a couple of messages, and watch your messages coming in on your kara browser tab.

You can find another script example in our documentation (mind that this documentation example is using another input endpoint so you might need to make some adjustments) or take a look at our GitHub for even more examples.

You are free to use any of the CloudEngine capabilities (MQTT client, MQTT broker, Azure Event Hub or IoT Hub, …). Mind that some of our modules, such as SMS, requires additional asset registration on the API Solutions marketplace.

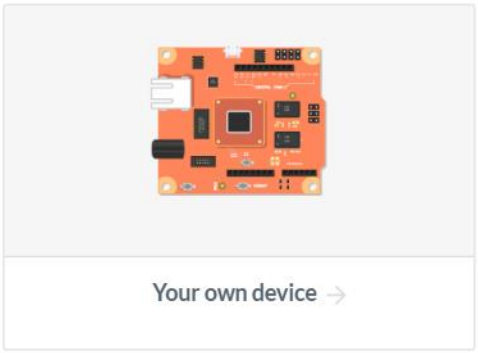
Sensitivity: Unrestricted

4 Note on AllThingsTalk Maker

The Proximus API Solution platform and the Proximus CloudEngine do not provide a standard integration module with the AllThingsTalk Maker solution. The Proximus MyThings connectivity solution proposed by AllThingsTalk will not work with LoRa4Makers.

If you wish to send your sensor data to the AllThingsTalk maker platform, you can:

- Create your device in a Maker Playground as “Your own device”



- Create assets for your new device, corresponding to the sensors you are using in your scenario. For instance, you could create an asset “temperature” of type “number”.
- Once the device and its assets are created, use the HTTP API or MQTT integration methods, as documented on the AllThingsTalk documentation website. The easiest method is most probably to use MQTT.

Here below is an EnCoScript example using MQTT, showing how an asset called “temperature” on AllThingsTalk Maker would be updated:

```
# Test ATT MQTT integration
object tempJson = json.createNewObjectNode();
tempJson.set("value","26.3");
object mqtt = create("MQTT", "tcp://api.allthingstalk.io",
"asset/your_asset_ID_from_ATT_maker/state");
mqtt.setUsername("maker:your_allthinstalk_device_token");
mqtt.setPassword("whateverpassword");
mqtt.setPayload(tempJson.getString());
object mqqtSend = mqtt.send();
```

Sensitivity: Unrestricted